REMARKS

Reconsideration and allowance of the subject application are respectfully requested.

Claims 1-18 stand rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent 5,136,691 to Baror. This rejection is respectfully traversed.

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.*, 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.*, 793 F.2d 1565 (Fed. Cir. 1986). Baror fails to satisfy this rigorous standard.

Baror teaches supporting caching of interlock (analogous to semaphore) variables in cache memory units in a multiprocessor and/or multitasking environment. Baror allows for exclusive access to shared memory by implementing atomic, read-modify-write transactions and by using a memory lock.

The Examiner relies on columns 39, 49, and 50 of Baror, which disclose how read and write operations of interlocked variables are implemented. Column 48, lines 16-17 states that only one processor is allowed to access interlock variables at any given time. An interlocked read is treated as a cache miss unless it hits a shared block in cache. In the event of a hit, the data is supplied by the cache with no memory bus access. But, in

the event of a cache miss, the interlocked variable is loaded from main memory with an *MLOCK signal asserted. (See column 12).

The interlocked write operations fall into two categories:

a) a STOREL write interlocked: occurs if the *LOCK bit (indicating that processor cache access is to an interlocked variable-see column 9) was set in the cycle preceding the write access;

b) a LOADSET write interlocked: occurs if the *LOCK bit was not set in the cycle preceding the write access.

The STOREL instruction is used to release an interlocked variable. The LOADSET instruction causes a sub-block including the interlock variable to be loaded into cache and is used to test and set an interlock variable. If the interlock variable value is 0, then it (or the area it protects) can be used by the processor; whereas, if the value is 1, it cannot be used. The first LOADSET read instruction reads data from memory and places it in cache, and the first LOADSET write involves a write memory. For both STOREL and LOADSET interlocked writes, a *MLOCK output is asserted. If a different master cache performs a memory write with *MLOCK asserted, then all other caches invalidate their own copy of the interlocked variable.

Amended claim 1 incorporates the subject matter of original claims 2, 4, and 5. In short, Baror does not disclose that if it is determined that the semaphore identifying data has been cleared between the step of retrieving the semaphore value from the semaphore value store and the step of writing the new semaphore value (indicative of exclusive

access having been granted), then the step of writing the new semaphore value is not attempted. Hence, Baror fails to anticipate claims 1, 18, and 19.

The Examiner asserts that column 49, lines 30-32 of Baror discloses the subject matter of claim 5. Applicant disagrees. Baror discloses in column 49, lines 9-38, that a LOADSET write operation causes the interlocked variable to be loaded into cache and is used to test and set an interlock variable. Accordingly, when the interlocked variable is loaded into cache, the variable value should be changed from 0 (indicating that the variable can be used by the processor) to the value 1 (indicating that the interlocked variable cannot be used). Only if a LOADSET read operation generated a cache miss will the data be written to main memory as well as to cache (see column 49, lines 29-34).

The *MLOCK output is asserted during the memory write that occurs as a consequence of the LOADSET write operation (see column 49, lines 35-36). If a different master cache performs a memory write with *MLOCK asserted, then all other caches invalidate their own copy of the interlocked variable. The Examiner asserts that such invalidation of the cached interlocked variable value corresponds to the step of clearing the stored semaphore identifying data. Accordingly, Baror discloses that the semaphore identifying data is cleared as a *direct consequence* of a LOADSET write operation that follows the reading of the semaphore value from the semaphore value store (main memory), i.e., as a direct consequence of a cache miss.

Indeed, Baror teaches that the data (i.e., new semaphore value) is not written in memory. In other words, writing of the new semaphore value to memory is <u>not</u>

attempted by Baror if there has been a cache hit in the read access of the LOADSET instruction, even if it is still written to cache. If there has been a cache hit, then the *cached value* of the interlocked variable (which is the counterpart of the semaphore identifying data of claim 1) certainly cannot have been cleared (invalidated). In fact, in Baror, the new semaphore value is written to memory as well as to cache only if the LOADSET read instruction generated a cache miss. Thus, Baror teaches away from the feature of the claim 1 whereby the writing of a new semaphore value (indicative of exclusive access being granted) it is not attempted if the semaphore identifying data (stored in cache in the system of Baror) has been cleared.

Claims 1-13 and 15 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent 6,446,149 to Moriarty. This rejection is respectfully traversed.

Like Baror, Moriarty does not disclose the claim feature where if it is determined that the semaphore identifying data has been cleared between the step of retrieving the semaphore value from the semaphore value store and the step of writing the new semaphore value, then the step of writing the new semaphore value is not attempted. Hence, Moriarty fails to anticipate claims 1, 18, and 19.

Moriarty discloses a synchronization memory address space for communication between multiple busmasters in a computer system. The address space includes a set of semaphore memory cells mapped to shared critical resources in the computer system. Ownership of a semaphore memory cell is achieved by a busmaster using a single operation, rather than at least two operations comprising a read and subsequent write or

set operation. This is achieved by semaphore memory cells that can switch themselves from an idle state to a busy state *responsive to* a first read operation by the busmaster. Thereafter, the busy state is communicated to any other busmasters that attempt to read the busy semaphore memory cell. The semaphore memory cell can be switched back from the busy state to the idle state in response to a write operation by the busmaster that currently owns the cell.

The Examiner asserts that column 9, lines 26-29 of Moriarty discloses the feature of claim 1 whereby a new semaphore value is written to the semaphore value store indicative that exclusive access to the processing resource has been granted. This passage in Moriarty specifies that when a CPU performs a read operation on the semaphore memory cell, the cell switches itself from an idle state to a busy state.

The Examiner further asserts that column 12, lines 35-37 of Moriarty discloses the subject matter of original claim 5. As shown in Figure 13 of Moriarty, a busmaster can be used to cache the value of a semaphore memory cell in that system. In Moriarty, the counterpart of the semaphore value store is the semaphore memory cell, and the counterpart of the semaphore identifying data is the cache semaphore data. Column 12, lines 35-37 of Moriarty simply discloses how certain write operations are used to place a cache line that corresponds to a semaphore memory cell on which a write operation has been performed in an invalid state. Thus, column 12, lines 35-37 discloses that when a new semaphore value has been written to a semaphore value store, the semaphore identifying data is ***subsequently*** invalidated, (i.e., cleared). Moriarty does not disclose

that if the cached value has been cleared *between the step of retrieving the semaphore value from the memory cell and the step of writing the new value to the memory cell,* then writing of the new semaphore value is not attempted. In fact, Moriarty teaches away from this feature by specifying that the semaphore memory cell switches itself from an idle state to a busy state *responsive to* a read operation. (See column 2, lines 27-30).

The technical problem addressed by this application is to avoid inappropriately granting exclusive access to an exclusive access requestor where some event such as an interrupt, an exception, or a context switch has occurred after a semaphore value has been read, but before exclusive access has been stabilized by writing of a new semaphore value. This problem is solved by not attempting to write a new semaphore value if the semaphore identifying data has been cleared between the step of retrieving the semaphore value from the semaphore value stored and the step of writing a new semaphore value indicative that exclusive access has been granted. Avoiding an inappropriate write may also save bus resources within a system having a shared bus by which the write action must be performed. For example, if the write action will use the main memory, which is typically on a shared bus, then stopping this write action saves bus resources. Freeing the shared bus for use by other parts of the system increases the overall efficiency of the system.

As explained above, both Baror and Moriarty teach away from what is claimed. Baror implements an atomic read-modify-write system in which a memory lock is implemented during write operations for semaphore (interlocked) variables. Moriarty

provides semaphore memory cells operable to respond to a first read operation by a busmaster by automatically switching from an idle state to a busy state.

This application is condition for allowance. An early notice to that effect is earnestly solicited.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: _____
John R. Lastova
Reg. No. 33,149

JRL:at
1100 North Glebe Road, 8th Floor
Arlington, VA 22201-4714
Telephone: (703) 816-4000
Facsimile: (703) 816-4100